An Empirical Investigation of Software Reliability Indicators

Sharifah Mashita Syed Mohamad (B.IT., M.CS.)

A dissertation submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Faculty of Engineering and Information Technology

University of Technology, Sydney

February 2012

CERTIFICATE OF AUTHORSHIP

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

Acknowledgement

I would like to express my deepest and sincerest gratitude to my supervisors. Dr. Tom McBride, I greatly appreciate the encouragement, guidance and generous sharing of ideas and profound knowledge in software engineering throughout the research process. Thank you for your time and effort in assisting me. Professor Didar Zowghi, I admire your sharp argumentation and attention to details. Your guidance and insight into the whole process were truly invaluable.

Next, to my beloved husband, Syed and son, Arfaan, thank you for being very understanding, supportive and for all the sacrifices both of you did for me. I am forever indebted to my parents for their unconditional love, endless patience and caring support. I thank my parents-in-law too for their love and help. I greatly appreciate the logistical support given by my brother, Mahadi during the field work. To Dr. Sharifah Mastura, thank you for your moral support and endless inspiration. A special thanks to Athira for her help in the thesis writing. There are no words to express my gratitude for all the prayers from all family members.

This thesis would not been possible without the willingness and enthusiasm shown by all the participants involved in this project. Thank you for all of the trust and opportunity in conducting this research project. Thank you also to the operation manager and his team at the telecommunications company for providing us the defect data.

I am grateful to my thesis editor, Mr. Jules Baumann, for his great efforts in proofreading my work. I also would like to express my gratefulness to Dr. Richard Raban, the head of School of Software, and friends of Human Centered Technology Design (HCTD) in University of Technology, Sydney.

My appreciation goes as well to my sponsors, The Ministry of Higher Education, Malaysia and University of Science, Malaysia (USM). My thanks also go the academics at USM; Professor Rosni from the School of Computer Science, Professor Hamidi and Professor Badorul, both from the School of Civil Engineering. Finally, not to be missed, Dr. Kamal Zuhairi from the School of Electrical and Electronic Engineering for sharing his opinion on software testing.

Table of contents

List of figures

List of tables

Terminology

Commercial-Off-The-Shelf (COTS)	A commercial software product that is developed for the general market, that is to be used by large numbers of people with a variety of users in different scenarios (Yang *et al.*, 2005 ; Boehm, 2006 ; Bishop *et al.*, 2007).
Commercial software	Proprietary software that is developed using in-house commercial development processes.
Open source software	In contrast to commercial software, open source software refers to software that is developed through open source software development. Open source software is software for which the source code is made freely available for others to access, use, copy, modify and redistribute (Raymond, 2000). This term is discussed in details in Section 2.4.1.
Open source software development	A collaborative way of producing software by widely dispersed developers collaborating over the Internet (Stallman, 1999 ; O'Reilly, 1999 ; Raymond, 2000 ; Scacchi, 2002 ; Scacchi et al., 2006 ; Crowston and Annabi, 2007 ; Wu et al., 2007). The Open Source Initiative (2006) describes it as "a development method for software that harnesses the power of distributed peer review and transparency of process" among the open source community.
Periodically stabilised software	A term used to refer to software that does have a stabilisation phase in its development process. This term is introduced as a result of the investigation of this research and discussed in Section 2.4.4.
Rapidly evolving software	A term used to refer to software that does not have a stabilisation phase in its development process. A characteristic of this development method is that new features and functionality are constantly introduced at the same time that defects are being rectified. This term is introduced as a result of the investigation of this research and discussed in Section 2.4.4.

Software reliability growth	Software reliability growth is defined as an improvement in the reliability over a period of time (Lyu, 1996). Growth in reliability usually happens through freezing the functionality at a certain stage during system testing, and then fixing defects as they are detected. The rate of detection and fixing of outstanding defects and the overall decline in the number of outstanding defects with respect to time or (testing effort) indicates the level of reliability. A growth in reliability signs that the software had reached an adequate level of reliability before being deployed. It is observed during a stabilisation phase.
Stabilisation phase	A stabilisation phase is a time period where the software does not have any more features added (i.e. "feature freeze"). This feature freeze is enforced during which system testing is conducted and all the defects that are found are repaired. As a result of this testing-debugging process, a definite stabilising trend in the defect modelling can be observed - which indicates a growth in reliability of software. The phase is long enough to observe the growth and decline of exposed defects. Obviously, system testing will continue until no failure can arise from the defect fix. This term is discussed in Section 2.3.2.
Test-first	Test-first is a programming approach where software developers write unit tests (including functional tests) before writing the code. Case studies in the literature often use the term "test-first" when making comparison with the test-last programming approach (Siniaalto and Abrahamsson, 2007 ; Dybå and Dingsøyr, 2008 ; Nagappan *et al.*, 2008). So, for the purpose of this research, the term 'test-first' is used throughout this thesis.
Test-last	In contrast to test-first, test-last is a "code and then test" programming approach as typically done in software development.
Test rigour	Test rigour is concerned with effectiveness and efficiency of testing. The term "effective testing" normally means the capability of finding defects, whereas "efficient testing" is defined as the capability of finding defects in a timely and cost effective manner (Bertolino, 2007). Test rigour is defined in Section 4.3.
Test coverage	A measure of how well a test suite tests a program. Test coverage can be measured in terms of code, requirements and design

	coverage. Test coverage is defined in Section 4.3.
Test sufficiency	Test sufficiency covers the criteria for test completion or stopping criteria and release software. Test sufficiency is defined in Section 4.3.

Abstract

This thesis investigates how individuals and organisations, without technical skills, might determine the reliability of open source software given the increased use of such software. Software reliability is normally indicated by the growth and subsequent decline of defects in the software. A notable observation is that reliability growth models require a definitive stabilisation phase during which testing can reveal the growth and decline of defects as the indication of the increase in reliability of software. However, there is not necessarily a definitive stabilisation phase in the open source software development. More importantly, the presence or absence of the stabilisation phase is an attribute of a software development method and is not restricted to open source software. When software is developed without a definitive stabilisation phase, reliability growth models are not applicable because the conditions for their validity have not been achieved. Consequently, this thesis looks for alternative information based on tests to aid decision-making about software acquisition. Data was collected by conducting semi-structured interviews from 29 participants who were currently engaged in software development. The information of tests; coverage, sufficiency and rigours of tests concerns the testing that has been performed on the software product and gives expectations on how well the software product has been tested.